Express Mail No. EL821158227US
Docket No. STL920000095US1
Firm No. 0055.0040

METHOD, SYSTEM, AND PROGRAM FOR USING OBJECTS IN DATA STORES DURING EXECUTION OF A WORKFLOW

RELATED APPLICATIONS

[0001] This application is related to the following copending and commonly assigned patent filed on the same date herewith, which are incorporated herein by reference in their entirety:

"Method, System, and Program for Generating a Workflow", having attorney docket no. STL920000094US1; and

"Method, System, and Program for Executing a Workflow", having attorney docket no. STL920000099US1.

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] The present invention relates to a method, system, and program for using objects in data stores during execution of a workflow.

2. Description of the Related Art

[0002] A workflow program allows businesses and other organizations to define their business operations as a computer model known as a workflow. A workflow defines a series of processes to be performed by users at a client computer. The user activities at the client computers may involve updating an electronic form, reviewing information, etc. After one user in the workflow performs a specified action, the work item or other information is then routed to one or more further nodes where further action may be taken. For instance, an on-line purchase of a product may involve numerous steps, such as receiving the customer order, routing the customer order to the credit department to process the bill and then routing the order to the shipment department to prepare the shipment. Once the shipment is prepared, the product may be shipped and information

on the purchase is then transferred to the customer service department to take any further action. Each of these processes may be defined as nodes in a workflow. A workflow program would then route the customer order to the business agents designated to handle the job. For instance, the initial order would be received by the order department and then routed to a person in shipping and billing. Once the bill and package are prepared, a further invoice may be forwarded to shipping. After shipping sends the package, the shipping agent may then enter information into the invoice and forward the electronic invoice to customer service for any follow up action.

[0003] A workflow is designed using workflow software, such as the International Business Machines (IBM) MQSERIES** Workflow software product. A process modeler is a person that analyzes the business operations, determines how the information related to the operations is routed electronically to client computers, and then defines a workflow model of the operations. The workflow model may be coded in the FlowMark Definition Language (FDL). The workflow model is then imported into a Runtime program that verifies and translates the workflow model into a process template. An instance of the process template can then be invoked to automates the sequence of events defined by the model.

[0004] There is a continued need in the art to provide improved techniques for building and utilizing workflow models.

SUMMARY OF THE PREFERRED EMBODIMENTS

[0005] Provided is a method, system, and program for implementing a workflow comprised of nodes. A workflow packet is generated that is accessed by users at the nodes in the workflow. A request is received to add one target object in one of the data stores to the packet, wherein each data store includes multiple objects. A determination is made of a first object identifier of the target object that is used to identify the target object in one data store. A second object identifier is generated indicating the data store including the target object and the first object identifier. The generated second object

identifier is inserted into the workflow packet, where nodes accessing objects in the workflow packet use the second object identifier to access the target object for use at the node.

[0006] In further implementations, the data stores are capable of being different types of data stores and from different vendors

[0007] Further provided is a method, system, and program for performing an Input/Output (I/O) operation on an object during execution of a workflow comprised of nodes. A plurality of objects are stored in one of multiple data stores, wherein each object is identified within the data store with a first object identifier. A workflow packet references at least one object with a second object identifier, wherein the second object identifier indicates one of the data stores and the first object identifier of the referenced object in the data store. An I/O request for one target object referenced by one second object identifier in the workflow packet is received from one node. A determination is made from the second object identifier of the data store for the target object and the first object identifier of the target object. The I/O request is performed on the target object at the determined first object identifier in the determined data store.

[0008] In further implementations, one workflow packet is associated with a plurality of the nodes in the workflow, wherein the nodes submit I/O requests for one or more objects referenced by the second object identifiers in the workflow packet.

[0009] In still further implementations, the I/O request comprises a workflow I/O request and each data store is accessed using data store interfaces. Provided is a mapping of workflow I/O requests to the data store interfaces for each of the data stores, wherein each workflow I/O request maps to one or more data store interfaces for each data store. Each data store interfaces for each data that are capable of implementing the workflow I/O request in the data store. A determination is made from the mapping the at least one data store interface for the determined data store that implements the workflow I/O request in the determined data store. The workflow I/O request is performed using the determined at least one data store interface.

[0010] Still further, the data stores are capable of comprising different types of data stores from different vendors that utilize different sets of data store interfaces to enable access to objects in the data store. In such case, the mapping enables the workflow I/O requests to be executed across heterogeneous data stores.

[0011] The described implementations provide a technique for implementing references to objects dispersed across multiple data stores to allow nodes in a workflow to access the objects in any of the data stores. The described techniques are capable of allowing nodes access to the objects in the different data stores even when the data stores comprise heterogeneous data stores of different types and from different vendors.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

FIG. 1 illustrates a workflow computing environment in which aspects of the invention are implemented;

FIGs. 2-7 illustrate graphical user interface (GUI) panels used to design a workflow model in accordance with implementations of the invention;

FIG. 8 illustrates logic implemented in a buildtime program to generate a worfklow model and worfklow definition language (WDL) file in accordance with implementations of the invention;

FIG. 9 illustrates logic performed by a workflow server to execute a workflow in accordance with implementations of the invention.

FIG. 10 illustrates a computing environment including multiple data stores storing objects in accordance with implementations of the invention;

FIG. 11 illustrates a format implementation of the object identifier in accordance with implementations of the invention;

FIG. 12 illustrates further details of components used to access objects from multiple data stores in accordance with implementations of the invention;

FIG. 13 illustrates logic to add objects to a workflow packet in accordance with implementations of the invention; and

FIG. 14 illustrates logic to access objects in a workflow packet in accordance with implementations of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0013] In the following description, reference is made to the accompanying drawings which form a part hereof and which illustrate several embodiments of the present invention. It is understood that other embodiments may be utilized and structural and operational changes may be made without departing from the scope of the present invention.

[0014] FIG. 1 illustrates a workflow environment implementation in which the invention is realized. A workflow engine 2 includes a runtime database 4 and a workflow server 6, such as the IBM MQSeries Workflow server. The workflow server 6 is capable of transforming a workflow model coded in a workflow definition language (WDL) file 10, such as FDL, into a process template 8 implemented in the runtime database 4. The runtime database 4 stores database tables that implement the data structures that provide the status and setup information needed for workflow process execution. Whenever the state of a process activity changes, such information is recorded in the runtime database 4. The runtime database 4 may be implemented using any database program known in the art, such as IBM DB2.**

[0015] The workflow server 6 coordinates and manages the execution of processes for a defined process template 8. The workflow server 6 executes any programs associated with a process defined for the workflow, interprets the process definitions, creates process instances and manages their execution, manages processes and states, logs events, communicates with users as part of the workflow, etc. The workflow server 6 may include a database client program (not shown) to access and update records related to the

workflow being processed maintained in the runtime database 4. The workflow server 6 processing may be distributed across multiple computers to achieve workload balancing.

[0016] The workflow clients 12a, b...n represent the client computers that execute workflow application program interfaces (APIs) to perform workflow related actions and activities and return messages to the workflow server 6. The workflow clients 12a, b...n thus comprise instances of the workflow code on the client computers that allow users to interface with the executing workflow and the workflow server 6. The workflow server 6 would execute activity programs as part of executing the workflow and transmit messages and data to the workflow client 12 to request user action to proceed with the workflow. The actions associated with the nodes and executed by the workflow server 6 may comprise Java servlets. The workflow client 12 may comprise a Web browser capable of executing Java scripts transferred from the Java servlet executing on the workflow server 6. Further, details on implementations and interactions of the workflow server 6 and client 12 are described in the IBM publication "IBM MQSeries Workflow: Concepts and Architecture, Version 3.3", IBM document no. GH12-6285-03 (March, 2001), which publication is incorporated herein by reference in its entirety.

[0017] A workflow builder 20 comprises a system including a buildtime program 22 that implements a plurality of graphical user interface (GUI) panels in which a user may define the components of a workflow model 24. A workflow translator 26 converts the workflow model 24, with the defined workflow components, into a workflow definition language (WDL) file 10 that implements the workflow model 24. The workflow definition language (WDL) may comprise the FlowMark Definition Language (FDL), Workflow Process Definition Language (WPDL) or any other workflow definition language known in the art that is used to define workflows. The workflow translator 24 would transfer the WDL file 10 to the workflow server 6 to transform into a process template 8 in the runtime database 4 in a manner known in the art.

[0018] The workflow engine 2, and each of the program components therein, such as the runtime database 4 and workflow server 6, may be implemented in one or more

computing machines. The workflow clients 12 which provide the workflow interface to users may be implemented on one or more client machines. The workflow builder 20, including the buildtime program 22 and workflow translator 26 programs, may be implemented on one or more computing machines. Any portion of the workflow engine 2, workflow builder 20, and/or workflow client 12, and program components therein, may be implemented on the same computing machines or separate machines. The computing machines used to implement the workflow engine 2, workflow clients 12, and workflow builder 20 may comprise any computing device known in the art, such as a server, workstation, mainframe, personal computer, laptop computer, hand held computer, telephony device, etc.

[0019] As discussed, the buildtime program 22 generates a series of graphical user interface (GUI) panels through which the user may define a workflow. Before utilizing the buildtime program 22, the process modeler would plan the workflow and analyze the work the business performs, how it is performed, and by whom. The process modeler may then develop a workflow to generate a final product, which may comprise the result of the effort of a single business unit or the cumulative efforts of multiple users and units within an organization.. To produce the final product, a workflow packet comprised of one or more documents would transfer through various user work stations in the company defined as nodes in the workflow to require the user associated with such node to handle and process and forward to another user to handle. A document is comprised of a multimedia item that has digital content.

[0020] For instance, an insurance company may have to process numerous documents related to an insurance claim, such as photographs, appraisals, expert reports, etc. Employees may spend a substantial amount of time sorting through documents and associating the documents with particular claims. In the workflow model, all the documents related to a single claim would be part of a work packet that may move through various user stations to review and process. The workflow would comprise the

flow of work and actions that are performed on the documents or workflow packet by multiple users in the system.

[0021] The workflow defines the sequence and boundaries of how the work is performed with respect to the documents in the workflow packet, and any restrictions on the order in which documents in the workflow packet must be processed. For instance, before the claim can proceed to a further step, a claims adjuster might be required to ensure that certain documents are included in the workflow packet for the claim before the workflow packet can proceed to further nodes in the workflow, e.g., determining the amount of compensation.

[0022] In workflow terminology, a worklist is a queue of work items. Each work item comprises a unit of work for a node in the workflow that is performed by the users associated with that node. Each work item may be associated with one work packet, which comprises documents or objects that are processed during the work defined for that work item. When a user at one node accesses the work item to perform the work defined therein, that workitem is locked, thereby preventing others at that node from accessing the work item.

[0023] A worklist, which is a queue of work for the users of the organization to perform with respect to the workflow packet. The work items within the worklist can be handled by any of the employees/users assigned to the worklist. An action list defines the actions that a user can perform on the work packet objects associated with the work item, such as selections or data that may be entered in the work packet. For example, an adjuster in the claim process workflow can select an option to continue consideration of the claim if it appears valid or select an option to reject the claim. The workflow further consists of the paths defined as the connections between nodes which indicate the order of execution of nodes in the workflow.

[0024] FIG. 2 illustrates an example of a GUI panel 50 displayed by the buildtime program 22 illustrating a workflow 52 defined by a process modeler using workflow icons available in the GUI panel 50. The workflow 52 has a start icon 54, displayed as

an icon having a green light, that indicates the start of the workflow and an end icon 56 is defined as indicating an end of the workflow. The start 54 and end 56 icons may be added automatically to the workflow 52 when the process modeler begins working on the workflow model 24. Between the start 54 and end 56 icons are a plurality of work nodes 58, 60, and 62, displayed as three stacked envelopes, that associate work items on the worklist and an action list for a specific point in a workflow 52. A work node 58, 60, 62 is a point in the workflow where work is performed. A user exit icon 64 indicates a user exit node where an application program is called to perform a background operation. Certain user exits may require that the called application program provide data to a work item in the workflow. Alternatively, the user exit may call an application program that may execute in the background, such as update a database, print reports, etc., while the workflow proceeds to further nodes.

[0025] As mentioned, the start 54 and end 56 icons may be automatically added to the workflow 52 when the user starts a blank workflow model 24. The user may move the start 54 and end 56 icons to any location on the drawing surface. The user may select the control icons 66 and 68 to add work and user exit nodes, respectively, to the workflow. Control icon 70 is used to define a path between two nodes defining the sequence in which the nodes are processed, and the order in which the work items are processed. The path lines, which are shown as the arrows between the start 54, end 56, work and user exit icons 58, 60, 62, and 64 illustrate the operation flow from one node to another.

[0026] Once the process modeler has defined the general layout of a workflow using the control icons 66, 68, 70, and 72, as shown in the workflow 52 in FIG. 2, the process modeler may then use additional GUI panels shown in FIGs. 3-7 of the buildtime program 22 to associate particular users, actions and work items with the nodes.

[0027] FIG. 3 illustrates a GUI panel 100 used to define properties for a new workflow being defined in the panel 50 of FIG. 1. A description field 102 includes a description of the workflow being defined. An action list 104 is a list the actions that can be invoked at the nodes in the defined workflow. The actions may comprise programs that are executed

at a particular node. In certain implementations, the actions comprise Java methods that the workflow server 6 executes when control proceeds to the node with which the method is associated. The program modeler would associate the actions in the action list with particular nodes. An access list 106 defines a mapping of users that can be assigned to nodes to perform the action associated with such node. Selection of the enable notification checkbox 108 causes a message to be sent to a specified user if the user associated with a node has not performed the action defined for the node within a specified time frame indicated in the deadline field 110.

[0028] After defining the properties for the new workflow and placing icons in the drawing area and path arrows to define the workflow, the user would then use the GUI panels shown in FIGs. 4-7 to associate actions and a user with each node, where the associated user performs the associated action when the workflow server 6 processes the node according to the workflow. FIG. 4 illustrates a start node panel 130 in which the process modeler defines the action and user associated with the start node 54 in the workflow, i.e., the first user that will perform the action with respect to the item at the beginning of the workflow. The process modeler would select an action from the action list in the action field 132 and specify the user to perform the action at the start node in the user field 134. The enable notification of deadline checkbox 136 may be selected to notify the user associated with the start node that a deadline has passed during which the user designated action for that node was not completed. Anyone, such as an administrator, user associated with node, or other user on the access list may receive the notification of the missed deadline for the start node 54.

[0029] FIG. 5 illustrates the property GUI panel 150 used to associate one or more actions and a user with one of the work nodes in the workflow, such as work nodes 58, 60, and 62 in FIG. 2. The work nodes defined by the user may comprise a decision point node, collection point node, document node, and assign value node. A decision point node causes the workflow to proceed along a branch of execution based on selection by the user or some other action taken by an external application called at a previous work

node. For instance, the path taken to the next node in the workflow may vary if the claim adjuster selects to reject the claim as opposed to approving the claim. A collection point node is a work node where certain documentation is gathered and added to the work packet. The collection node holds and manages work packages that cannot be processed completely until additional information is received. A document node represents a document in the workflow.

[0030] FIGs. 6 and 7 illustrates the property panels used to define a user exit node that calls an external application to perform operations and perhaps transfer data back to the node for further processing. A user exit is a point in the workflow execution where a user exit routine can be given control, and transfer data back and forth from the external application to the node upon the occurrence of a user-specified event. Further, the user exit node may call an external application program to perform background operations while the workflow proceeds to the next work node.

[0031] Using all the above described panels shown in FIGs. 2-7, the process modeler can design a workflow model specifying work nodes and the actions associated with the work nodes, as well as the paths between the work nodes. If two paths lead into a node, such as the case with work node 62 in FIG. 2, then the workflow will only proceed to the action specified for that next node once the actions associated with the two preceding nodes has completed. Before the user may use the buildtime program 22, the user must define the access control lists, users, user groups, actions, action lists and worklist in a manner known in the art using application programming interfaces (APIs). The worklist would provide those work items assigned to users, indicating the work items a user may access when performing actions at a node.

[0032] FIG. 8 illustrates logic implemented in the workflow builder 20 and workflow engine 2 to generate a workflow in the runtime database 4 that may be invoked and executed. Control begins at block 200 with the buildtime program 22 receiving defined work lists, access control lists, and action lists. The buildtime program 22 and GUI panels therein are then used to generate (at block 202) a workflow model 24 including a

plurality of nodes, such as shown in FIG. 2, and paths therebetween defining the order of execution of the nodes in the workflow. At block 204, the workflow translator 26 converts the workflow model 24 and the defined workflow, access lists, action lists, etc. into a WDL file coded using a workflow definition language known in the art and transfers the WDL file 10 to the workflow server 6. The workflow server 6 then builds the process template 8, including tables and other data structures, in the runtime database 4 that are used to implement the workflow model defined using the buildtime program 22.

[0033] In certain implementations, the workflow model 24 defined using the buildtime program 22 is document centric in that the actions performed at the node concern the processing of work packages that may comprise any content or object that is processed and routed through the workflow. FIG. 9 illustrates the logic performed by the workflow server 6 to execute the workflow logic generated using the buildtime program 22 GUI panel 50 shown in FIG. 2. When a user invokes a workflow stored in the runtime database 4, the workflow server 6 accesses (at block 300) the start node of the invoked workflow by interacting with the runtime database 4 in a manner known in the art. From the properties defined for that node, the workflow server 6 determines (at block 302) the actions and user associated with the node. The workflow server 6 further processes (at block 304) the access list defined for the workflow to determine the work item for the accessed node. If (at block 306) the determined work item currently accessed in the workflow is locked by another user at that node, then the workflow server 6 waits (at block 308) for the lock on the work item(s) to be released. If the work item is not locked or after the lock is released, control proceeds to block 310 where the workflow server 6 places a lock on the determined work item. The workflow server 6 then executes (at block 312) the action associated with the node and communicates data to the workflow client 12 of the determined user requesting user action.

[0034] If (at block 314) notification is enabled for the current node and the deadline has passed (at block 316) without receiving a response from the user, then the workflow

server 6 notifies the user specified with the enable notification that the deadline has passed. Upon receiving (at block 318) a response from the user, which may comprise entering information, modifying a work item, adding a work item to the work package, selecting an option, etc., the workflow server 6 unlocks (at block 320) the work item(s) previously locked for the user. If (at block 322) the current node is the stop node, then control ends; otherwise, if there are further nodes to process in the workflow, then the workflow server 6 determines (at block 324) from the path from the current node the next node in the workflow and accesses (at block 326) the next node. Control then proceeds back to block 326 to process the next node.

[0035] The workflow logic of FIG. 9 provides a document centric workflow in that the state of processing work items associated with the node controls the workflow because control cannot proceed to other subsequent nodes that process the locked work item until the node holding the lock completes execution and releases the lock on the work item. Thus, access to work items controls the flow through the workflow. The workflow builder 20 provides a GUI tool to allow the user to create a document centric workflow model and translate that workflow.model 24, including the defined worklists, access lists, action lists, etc., into a workflow definition language (WDL) file 10 that can be maintained and utilized in a robust workflow engine software product known in the art.

[0036] With the described implementations, the workflow builder 20 generates a WDL file 10 that may be compatible with workflow engines from different vendors because different vendors may design their workflow engines to be compatible with the WDL format of the WDL file 10. This allows the workflow model defined in the WDL file 10 to be transportable across different vendor workflow engine platforms.Utilizing Documents from Heterogeneous Data Stores in the Workflow

[0037] FIG. 10 illustrates a workflow architecture implementation in which the workflow can utilize documents from heterogeneous data stores. A workflow engine 402 may comprise the workflow engine 2 described above with respect to FIG. 1 for implementing a workflow. Workflow clients 412a, b, c may comprise the workflow

clients 12a, b, c described above with respect to FIG. 1. The workflow clients 412a, b, c interface with the data store interface 422 and execute workflow APIs 420 to perform workflow related actions and activities on the workflow engine 402 that manages the workflow. A data store interface 422 allows the workflow clients 412a, b, c to use workflow APIs 420 to access objects, such as documents, multimedia files, etc., from different data stores 432a, b, c. The data stores 432a, b, c may comprise different types of repositories from different vendors. For instance, the data stores 432a, b, c may comprise databases implemented through the database programs of different vendors, e.g., Oracle Corporation, etc., or a data store product, such as the IBM Content Manager** product that manages and integrates access to scanned images, facsimiles, electronic office documents, Extensible Markup Language (XML), Hypertext Markup Language (HTML) files, computer output, audio, video, etc.

[0038] Each of the data stores 432a, b, c may be accessed using data store APIs 430a, b, c provided by the particular data store vendor. For instance, if the data store 432a, b, c comprises a database, then the data store APIs 430a, b, c would comprise the database client APIs needed to request data from a database server managing access to the database. In the case of the database, the data store APIs 430a, b, c would comprise the data store vendor's particular implementation of the Structured Query Language (SQL).

[0039] The data store interface 422 further includes a workflow/data store API mapping 434 that provides a correspondence of the workflow APIs 420 to APIs in each of the data store APIs 430a, b, c. For instance, if the workflow API requests to OPEN a document, then the workflow/data store API mapping 434 would provide the corresponding OPEN command for one of the vendor data stores API 430a, b, c. The workflow/data store API mapping 434 may use any technique known in the art to map commands from the workflow APIs 420 to one or more commands in each of the data store APIs 430a, b, c.

[0040] In certain implementations, objects in the data stores 432a, b, c are provided a unique identifier across all the data stores 432a, b, c, such that the unique identifier can be used to identify one object in one of the data stores 432a, b, c, referred to herein as a

persistent identifier ("PID"). FIG. 11 illustrates a format of the PID 450 as including a data store identifier that provides a unique data store identifier (ID) 452 of one of the data stores 432a, b, c including the object identified by the PID. The data store ID may comprise a network address of the data store 432a, b, c, such as an alias name or an IP address. An object identifier (ID) 454 provides a unique identifier of the object identified by the PID within the data store 432a, b, c, identified in the first field 452. In this way the PID 450 provides sufficient information to allow access to a specific object in one of the data stores 432a, b, c.

[0041] FIG. 12 illustrates a workflow implementation utilizing the PIDs 500a, b, c in a workflow packet 502 to enable users at the nodes 504a, b, c, d to access objects 506a, b...n, 508a, b...n, and 510a, b...n in the data stores 432a, b, c, respectively, during execution of the workflow. The objects may comprise any type of file, including a document, multimedia file (e.g., video, audio, three-dimensional, image, graphic, etc.), database file, spreadsheet, or any other type of file known in the art. Each node 504a, b, c, d is associated with one work item 512a, b, c, d that provides the actions to perform at the node 504a, b, c, d. Each work item 512a, b, c, d includes a reference 514a, b, c, d to the workflow packet 502, indicating that one or more of the objects identified by the PIDs 500a, b, c in the referenced workflow packet 502 may be accessed and updated by a user at the node 504a, b, c, d. Thus, the user at the nodes uses the PIDs 500a, b, c included within the workflow packet 502 referenced at such nodes to access the objects 506a, b...n, 508a, b...n, 510a, b...n in the data stores 432a, b, c for use at the nodes 504a, b, c, d. In the described implementations, a user at a node 504a, b, c, d may modify the content of the workflow packet 502 by adding and removing PID 500a, b, c, d references.

[0042] FIG. 13 illustrates logic implemented within the data store interface 422 to add one object 506a, b...n, 508a, b...n, 510a, b...n in one of the data stores 432a, b, c to the workflow packet 502. Control begins at block 550 upon receiving a request to add the object. The request may be received by a process modeler constructing the initial workflow packet 502 or by a user at one of the nodes 504a, b, c, d modifying the

workflow packet 502 during workflow execution. The user requesting the modification may use workflow APIs 420 translated to data store APIs 430a, b, c to locate objects in the data stores 432a, b, c. The data store interface 422 then determines (at block 552) the data store ID 452 and the object ID 454 of the requested object 506a, b...n, 508a, b...n, 510a, b...n and generates (at block 554) a PID for the object to add by concatenating the determined data store ID and the object ID. The generated PID is then added to the workflow packet 502.

[0043] FIG. 14 illustrates logic implemented in the data store interface 422 to perform an Input/Output (I/O) operation on an object 506a, b...n, 508a, b...n, 510a, b...n identified by a PID. A user at one of the nodes may submit a workflow API for a PID to perform the requested action, such as read, write or open one the object identified by the PID. Control begins at block 570 upon receiving the I/O request for the PID. The data store interface 422 determines (at block 572) the data store 432a, b, c including the requested object from the data store ID 452 (FIG. 11) component of the received PID. The workflow/data store API mapping 434 is then processed (at block 574) to determine the API from the data store APIs 430a, b, c for the determined data store 432a, b, c that corresponds to the received workflow API. The data store interface 422 then uses the determined API to access the object 506a, b...n, 508a, b...n, 510a, b...n in the determined data store 432a, b, c having the ID in the object ID field 454 (at block 576).

[0044] The described implementations provide an architecture to allow the workflow clients 412a, b, c performing actions defined in a work item 512a, b, c, d at one node 504a, b, c, d in the workflow to seamlessly access any object in one of multiple heterogeneous data stores 432a, b, c, that may comprise different data store types, e.g., databases, data repositories, etc., from different vendors. The workflow client 412a, b, c uses the workflow APIs to perform I/Os with respect to a PID referencing an object in any of the data stores 432a, b, c and the data store interface 422 determines the API from the data store APIs 430a, b, for the determined data store 432a, b, c to use to execute the submitted data store APIs 430a, b, c at the data store 432a, b, c.

[0045] Further, with the described implementations, the content or documents referenced in the workflow packet at the nodes 504a, b, c, d may be modified by modifying the PIDs 500a, b, c listed in the workflow packet 502. This allows the content of the workflow packet 502 to be modified as the workflow packet 502 is being routed between the workflow nodes 504a, b, c, d.

Additional Implementation Details

[0046] The preferred embodiments may be implemented as a method, apparatus or article of manufacture using standard programming and/or engineering techniques to produce software or code. The term "article of manufacture" as used herein refers to code or logic implemented in a computer readable medium (e.g., magnetic storage medium (e.g., hard disk drives, floppy disks, tape, etc.), optical storage (CD-ROMs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, firmware, programmable logic, etc.). Code in the computer readable medium is accessed and executed by a processor. The code in which preferred embodiments are implemented may further be accessible through a transmission media or from a file server over a network. In such cases, the article of manufacture in which the code is implemented may comprise a transmission media, such as a network transmission line, wireless transmission media, signals propagating through space, radio waves, infrared signals, etc. Of course, those skilled in the art will recognize that many modifications may be made to this configuration without departing from the scope of the present invention, and that the article of manufacture may comprise any information bearing medium known in the art.

[0047] The workflow client and server may be implemented within any vendor workflow program known in the art.

[0048] In the described implementations, the actions were implemented as Java methods. Alternatively, the actions may be implemented in any programming language known in the art.

[0049] In the described implementations, particular icons were used to represent different information in the workflow, such as work nodes, exit nodes, etc. However, any icon design may be used to represent the workflow components. Further, additional graphical representations may be provided for different types of work nodes, e.g., collection work nodes, assign value node, decision point node, etc.

[0050] In the described implementations, the class architecture is implemented as an object oriented class architecture. Alternatively, non-object oriented programming techniques may be used to implement the described class architecture.

[0051] In the described implementations, the data stores comprised heterogeneous data stores managed by different types of data management programs and from different vendors. Additionally, the data stores may be of a same type and from a same vendor.

[0052] The foregoing description of the preferred embodiments of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended hereto. The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

**MQSeries, IBM, and DB2 are registered trademarks of International Business Machines Corp.